林轩田《机器学习基石》课程笔记11 -- Linear Models for Classification

作者：红色石头　　公众号：AI有道（id：redstonewill）

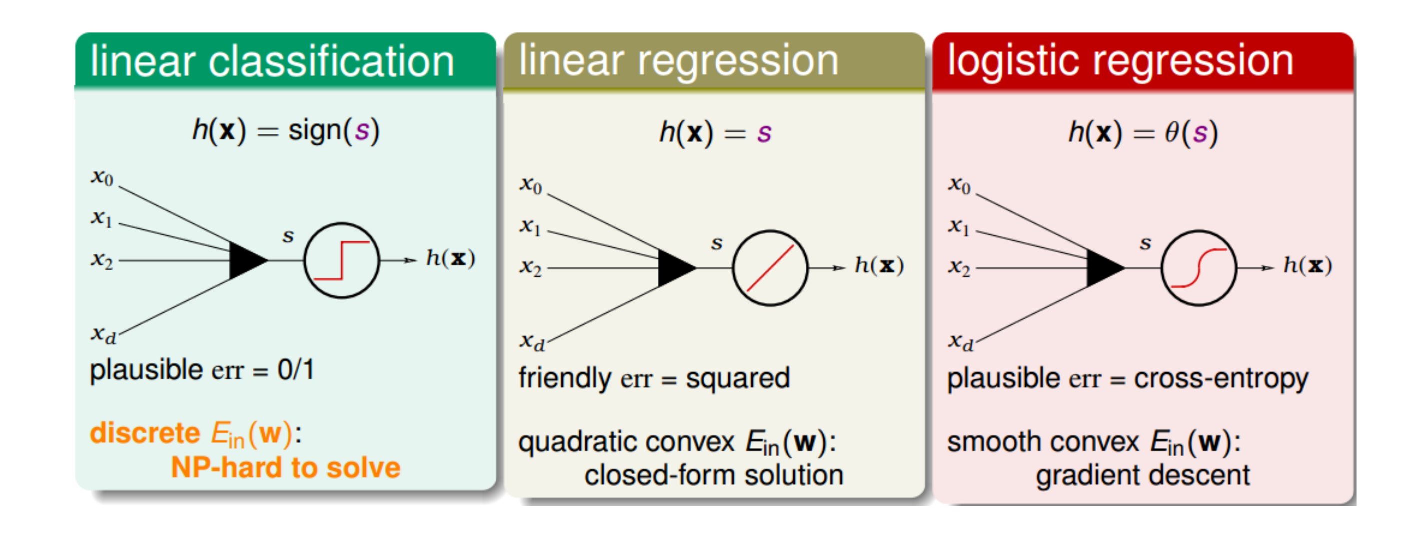
上一节课，我们介绍了Logistic Regression问题，建立cross-entropy error，并提出使用梯度下降算法gradient descnt来获得最好的logistic hypothesis。本节课继续介绍使用线性模型来解决分类问题。

一、Linear Models for Binary Classification

之前介绍几种线性模型都有一个共同点，就是都有样本特征x的加权运算，我们引入一个线性得分函数s：

$$s = w^T x$$

三种线性模型，第一种是linear classification。线性分类模型的hypothesis为$h(x) = sign(s)$,取值范围为{-1,+1}两个值，它的err是0/1的，所以对应的$E_{in}(w)$是离散的，并不好解，这是个NP-hard问题。第二种是linear regression。线性回归模型的hypothesis为$h(x) = s$，取值范围为整个实数空间，它的err是squared的，所以对应的$E_{in}(w)$是开口向上的二次曲线，其解是closed-form的，直接用线性最小二乘法求解即可。第三种是logistic regression。逻辑回归模型的hypothesis为$h(x) = \theta(s)$，取值范围为(-1,1)之间，它的err是cross-entropy的，所有对应的$E_{in}(w)$是平滑的凸函数，可以使用梯度下降算法求最小值。



从上图中，我们发现，linear regression和logistic regression的error function都有最小解。那么可不可以用这两种方法来求解linear classification问题呢？下面，我们来对这三种模型的error function进行分析，看看它们之间有什么联系。

对于linear classification，它的error function可以写成：

$$err_{0/1}(s,y) = |sign(s) \neq y| = |sign(ys) \neq 1|$$

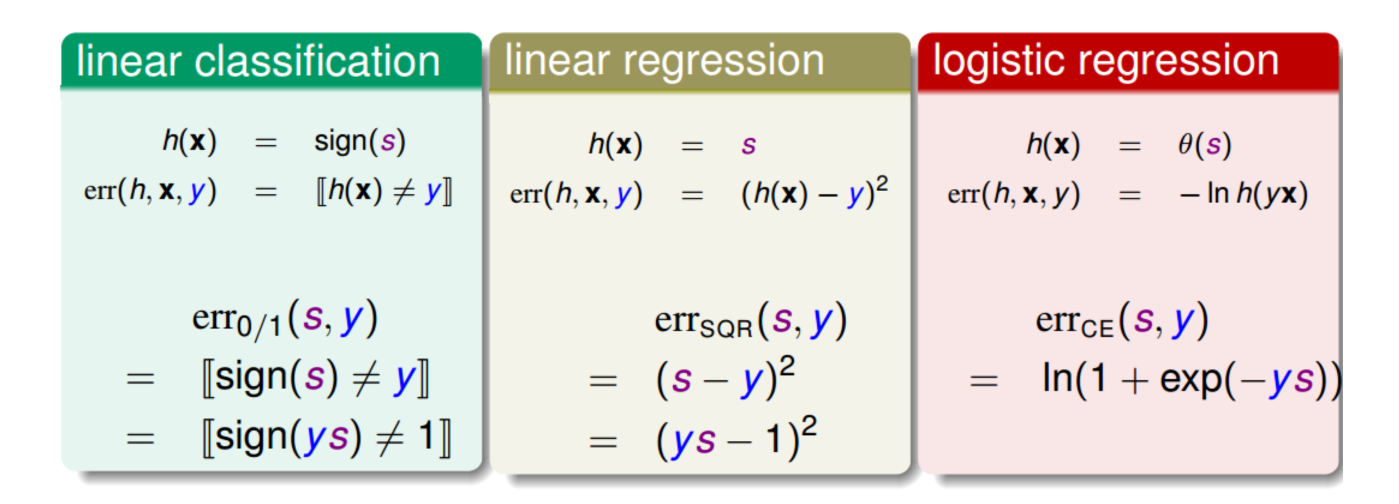
对于linear regression，它的error function可以写成：

$$err_{SQR}(s,y) = (s-y)^2 = (ys-1)^2$$

对于logistic regression，它的error function可以写成：

$$err_{CE}(s,y) = ln(1+exp(-ys))$$

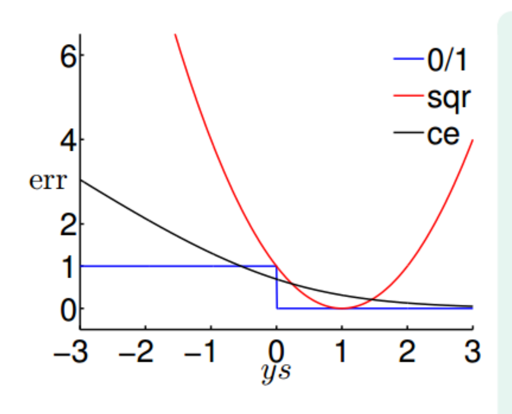
上述三种模型的error function都引入了ys变量，那么ys的物理意义是什么？ys就是指分类的正确率得分，其值越大越好，得分越高。



下面，我们用图形化的方式来解释三种模型的error function到底有什么关系：

Visualizing Error Functions

$$\begin{aligned}
\text{0/1} \quad & \text{err}_{0/1}(s,y) &=& [\![\text{sign}(ys) \neq 1]\!] \\
\text{sqr} \quad & \text{err}_{\text{SQR}}(s,y) &=& (ys-1)^2 \\
\text{ce} \quad & \text{err}_{\text{CE}}(s,y) &=& \ln(1+\exp(-ys)) \\
\text{scaled ce} \quad & \text{err}_{\text{SCE}}(s,y) &=& \log_2(1+\exp(-ys))
\end{aligned}$$

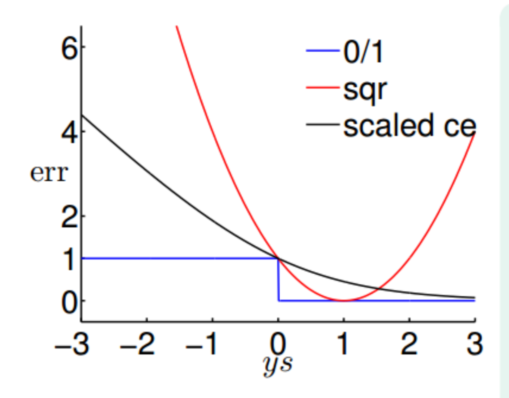


- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\text{err}_{\text{SQR}} \rightarrow$ small $\text{err}_{0/1}$
- ce: monotonic of ys
  small $\text{err}_{\text{CE}} \leftrightarrow$ small $\text{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
  small $\text{err}_{\text{SCE}} \leftrightarrow$ small $\text{err}_{0/1}$

从上图中可以看出，ys是横坐标轴，$err_{0/1}$是呈阶梯状的，在ys>0时，$err_{0/1}$恒取最小值0。err_{SQR}呈抛物线形式，在ys=1时，取得最小值，且在ys=1左右很小区域内，$err_{0/1}$和err_{SQR}近似。err_{CE}是呈指数下降的单调函数，ys越大，其值越小。同样在ys=1左右很小区域内，$err_{0/1}$和err_{CE}近似。但是我们发现err_{CE}并不是始终在$err_{0/1}$之上，所以为了计算讨论方便，我们把err_{CE}做幅值上的调整，引入$err_{SCE}=log_2(1+exp(-ys))=\frac{1}{ln2}err_{CE}$，这样能保证$err_{SCE}$始终在$err_{0/1}$上面，如下图所示：

Visualizing Error Functions

$$\begin{aligned}
\text{0/1} \quad & \text{err}_{0/1}(s,y) &=& [\![\text{sign}(ys) \neq 1]\!] \\
\text{sqr} \quad & \text{err}_{\text{SQR}}(s,y) &=& (ys-1)^2 \\
\text{ce} \quad & \text{err}_{\text{CE}}(s,y) &=& \ln(1+\exp(-ys)) \\
\text{scaled ce} \quad & \text{err}_{\text{SCE}}(s,y) &=& \log_2(1+\exp(-ys))
\end{aligned}$$



- 0/1: 1 iff $ys \leq 0$
- sqr: large if $ys \ll 1$
  but over-charge $ys \gg 1$
  small $\text{err}_{\text{SQR}} \rightarrow$ small $\text{err}_{0/1}$
- ce: monotonic of ys
  small $\text{err}_{\text{CE}} \leftrightarrow$ small $\text{err}_{0/1}$
- scaled ce: a proper upper bound of 0/1
  small $\text{err}_{\text{SCE}} \leftrightarrow$ small $\text{err}_{0/1}$

由上图可以看出：

$$err_{0/1}(s,y) \le err_{SCE}(s,y) = \frac{1}{ln2} err_{CE}(s,y)$$

$$E_{in}^{0/1}(w) \le E_{in}^{SCE}(w) = \frac{1}{ln2} E_{in}^{CE}(w)$$

$$E_{out}^{0/1}(w) \le E_{out}^{SCE}(w) = \frac{1}{ln2} E_{out}^{CE}(w)$$

那么由VC理论可以知道：
从0/1出发：

$$E_{out}^{0/1}(w) \le E_{in}^{0/1}(w) + \Omega^{0/1} \le \frac{1}{ln2} E_{in}^{CE}(w) + \Omega^{0/1}$$

从CE出发：

$$E_{out}^{0/1}(w) \le \frac{1}{ln2} E_{out}^{CE}(w) \le \frac{1}{ln2} E_{in}^{CE}(w) + \frac{1}{ln2} \Omega^{CE}$$

For any ys where $s = \mathbf{w}^T \mathbf{x}$

$$\text{err}_{0/1}(s, y) \le \text{err}_{\text{SCE}}(s, y) = \frac{1}{\ln 2} \text{err}_{\text{CE}}(s, y).$$

$$\Longrightarrow \quad E_{\text{in}}^{0/1}(\mathbf{w}) \le E_{\text{in}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w})$$

$$E_{\text{out}}^{0/1}(\mathbf{w}) \le E_{\text{out}}^{\text{SCE}}(\mathbf{w}) = \frac{1}{\ln 2} E_{\text{out}}^{\text{CE}}(\mathbf{w})$$

VC on 0/1:

$$E_{\text{out}}^{0/1}(\mathbf{w}) \le E_{\text{in}}^{0/1}(\mathbf{w}) + \Omega^{0/1} \le \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w}) + \Omega^{0/1}$$

VC-Reg on CE :

$$E_{\text{out}}^{0/1}(\mathbf{w}) \le \frac{1}{\ln 2} E_{\text{out}}^{\text{CE}}(\mathbf{w}) \le \frac{1}{\ln 2} E_{\text{in}}^{\text{CE}}(\mathbf{w}) + \frac{1}{\ln 2} \Omega^{\text{CE}}$$

small $E_{\text{in}}^{\text{CE}}(\mathbf{w}) \Longrightarrow$ small $E_{\text{out}}^{0/1}(\mathbf{w})$:
logistic/linear reg. for linear classification

通过上面的分析，我们看到err 0/1是被限定在一个上界中。这个上界是由logistic regression模型的error function决定的。而linear regression其实也是linear classification的一个upper bound，只是随着sy偏离1的位置越来越远，linear regression的error function偏差越来越大。综上所述，linear regression和logistic regression都可以用来解决linear classification的问题。

下图列举了PLA、linear regression、logistic regression模型用来解linear classification问题的优点和缺点。通常，我们使用linear regression来获得初始化的w_0，再用logistic regression模型进行最优化解。

PLA
- pros: **efficient + strong guarantee if lin. separable**
- cons: works only if lin. separable, otherwise needing **pocket** heuristic

linear regression
- pros: **'easiest' optimization**
- cons: loose bound of $\mathrm{err}_{0/1}$ for large $|ys|$

logistic regression
- pros: **'easy' optimization**
- cons: loose bound of $\mathrm{err}_{0/1}$ for very negative ys

- **linear regression** sometimes used to set $\mathbf{w}_0$ for **PLA/pocket/logistic regression**
- **logistic regression** often preferred over **pocket**

二、Stochastic Gradient Descent

之前介绍的PLA算法和logistic regression算法，都是用到了迭代操作。PLA每次迭代只会更新一个点，它每次迭代的时间复杂度是O(1)；而logistic regression每次迭代要对所有N个点都进行计算，它的每时间复杂度是O(N)。为了提高logistic regression中gradient descent算法的速度，可以使用另一种算法：随机梯度下降算法(Stochastic Gradient Descent)。

随机梯度下降算法每次迭代只找到一个点，计算该点的梯度，作为我们下一步更新w的依据。这样就保证了每次迭代的计算量大大减小，我们可以把整体的梯度看成这个随机过程的一个期望值。

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \underbrace{\frac{1}{N}\sum_{n=1}^{N} \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(y_n \mathbf{x}_n\right)}_{-\nabla E_{\text{in}}(\mathbf{w}_t)}$$

- want: update direction $\mathbf{v} \approx -\nabla E_{\text{in}}(\mathbf{w}_t)$
  while computing $\mathbf{v}$ by one single $(\mathbf{x}_n, y_n)$
- technique on removing $\frac{1}{N}\sum_{n=1}^{N}$:
  view as expectation $\mathcal{E}$ over uniform choice of n!

stochastic gradient:
$\nabla_{\mathbf{w}} \ \mathrm{err}(\mathbf{w}, \mathbf{x}_n, y_n)$ with random n
true gradient:
$\nabla_{\mathbf{w}} \ E_{\text{in}}(\mathbf{w}) = \underset{\text{random } n}{\mathcal{E}} \ \nabla_{\mathbf{w}} \ \mathrm{err}(\mathbf{w}, \mathbf{x}_n, y_n)$

随机梯度下降可以看成是真实的梯度加上均值为零的随机噪声方向。单次迭代看，好像会对每一步找到正确梯度方向有影响，但是整体期望值上看，与真实梯度的方向没有差太多，同样能找到最小值位置。随机梯度下降的优点是减少计算量，提高运算速度，而且便于online学习；缺点是不够稳定，每次迭代并不能保证按照正确的方向前进，而且达到最小值需要迭代的次数比梯度下降算法一般要多。

Stochastic Gradient Descent

- idea: replace true gradient by stochastic gradient
- after enough steps,
  average true gradient ≈ average stochastic gradient
- pros: **simple & cheaper computation :-)**
  —useful for **big data** or **online learning**
- cons: less stable in nature

对于logistic regression的SGD，它的表达式为：

$$w_{t+1} \leftarrow w_t + \eta\theta(-y_n w_t^T x_n)(y_n x_n)$$

我们发现，SGD与PLA的迭代公式有类似的地方，如下图所示：

SGD logistic regression:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \cdot \theta\left(-y_n \mathbf{w}_t^T \mathbf{x}_n\right)\left(y_n \mathbf{x}_n\right)$$

PLA:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + 1 \cdot \left[\!\left[y_n \neq \text{sign}(\mathbf{w}_t^T \mathbf{x}_n) \right]\!\right] \left(y_n \mathbf{x}_n\right)$$

我们把SGD logistic regression称之为'soft' PLA，因为PLA只对分类错误的点进行修正，而SGD logistic regression每次迭代都会进行或多或少的修正。另外，当$\eta = 1$，且$w_t^T x_n$足够大的时候，PLA近似等于SGD。

- SGD logistic regression ≈ 'soft' PLA
- PLA ≈ SGD logistic regression with $\eta = 1$ when $\mathbf{w}_t^T \mathbf{x}_n$ large

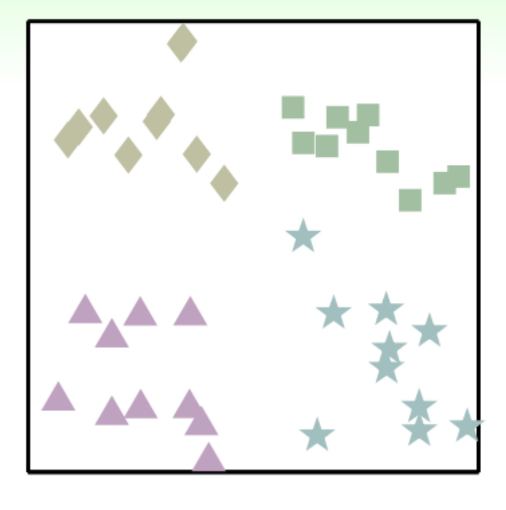
除此之外，还有两点需要说明：1、SGD的终止迭代条件。没有统一的终止条件，一般让迭代次数足够多；2、学习速率η。η的取值是根据实际情况来定的，一般取值0.1就可以了。

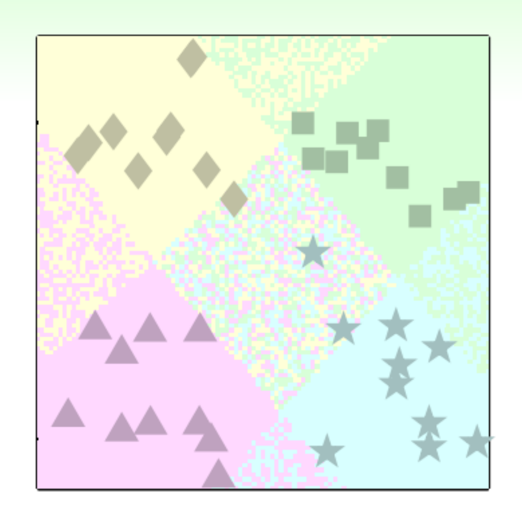
三、Multiclass via Logistic Regression

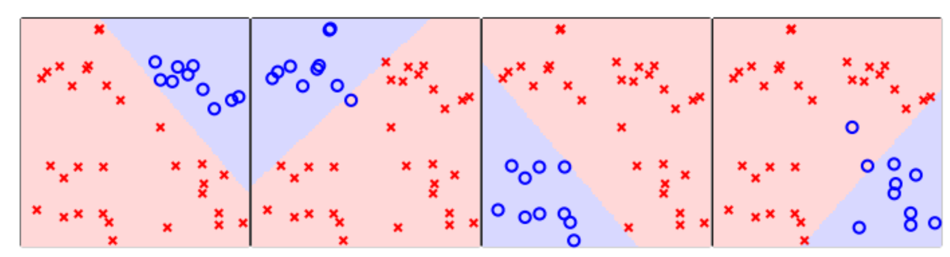
之前我们一直讲的都是二分类问题，本节主要介绍多分类问题，通过linear classification来解决。假设平面上有四个类，分别是正方形、菱形、三角形和星形，如何进行分类模型的训练呢？

首先我们可以想到这样一个办法，就是先把正方形作为正类，其他三种形状都是负类，即把它当成一个二分类问题，通过linear classification模型进行训练，得出平面上某个图形是不是正方形，且只有{-1,+1}两种情况。然后再分别以菱形、三角形、星形为正类，进行二元分类。这样进行四次二分类之后，就完成了这个多分类问题。

Multiclass Prediction: Combine Binary Classifiers





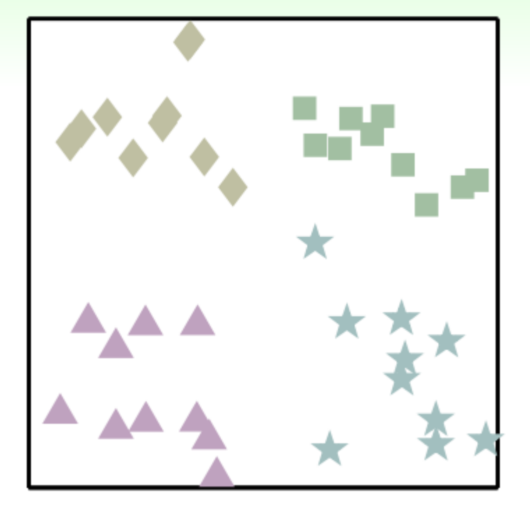


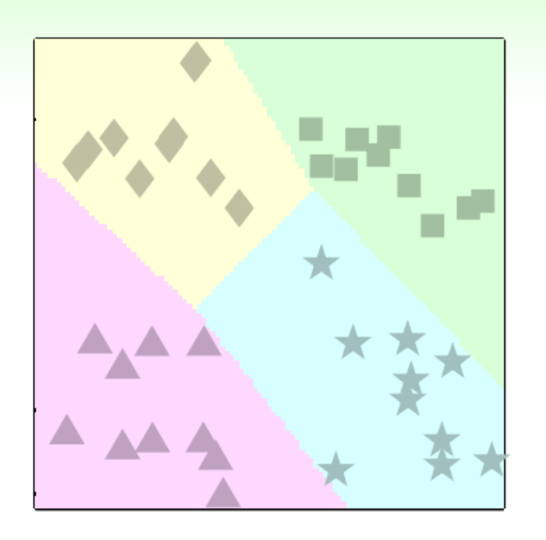
但是，这样的二分类会带来一些问题，因为我们只用{-1，+1}两个值来标记，那么平面上某些可能某些区域都被上述四次二分类模型判断为负类，即不属于四类中的任何一类；也可能会出现某些区域同时被两个类甚至多个类同时判断为正类，比如某个区域又判定为正方形又判定为菱形。那么对于这种情况，我们就无法进行多类别的准确判断，所以对于多类别，简单的binary classification不能解决问题。

针对这种问题，我们可以使用另外一种方法来解决：soft软性分类，即不用{-1，+1}这种binary classification，而是使用logistic regression，计算某点属于某类的概率、可能性，去概率最大的值为那一类就好。

soft classification的处理过程和之前类似，同样是分别令某类为正，其他三类为负，不同的是得到的是概率值，而不是{-1，+1}。最后得到某点分别属于四类的概率，取最大概率对应的哪一个类别就好。效果如下图所示：

Multiclass Prediction: Combine **Soft** Classifiers





$$g(\mathbf{x}) = \operatorname{argmax}_{k \in \mathcal{Y}} \theta\left(\mathbf{w}_{[k]}^{T}\mathbf{x}\right)$$

这种多分类的处理方式，我们称之为One-Versus-All(OVA) Decomposition。这种方法的优点是简单高效，可以使用logistic regression模型来解决；缺点是如果数据类别很多时，那么每次二分类问题中，正类和负类的数量差别就很大，数据不平衡unbalanced，这样会影响分类效果。但是，OVA还是非常常用的一种多分类算法。

One-Versus-All (OVA) Decomposition

1. for $k \in \mathcal{Y}$
   obtain $\mathbf{w}_{[k]}$ by running logistic regression on

$$\mathcal{D}_{[k]} = \{(\mathbf{x}_n, y'_n = 2 [\![y_n = k]\!] - 1)\}_{n=1}^{N}$$

2. return $g(\mathbf{x}) = \text{argmax}_{k \in \mathcal{Y}} \left(\mathbf{w}_{[k]}^T \mathbf{x} \right)$

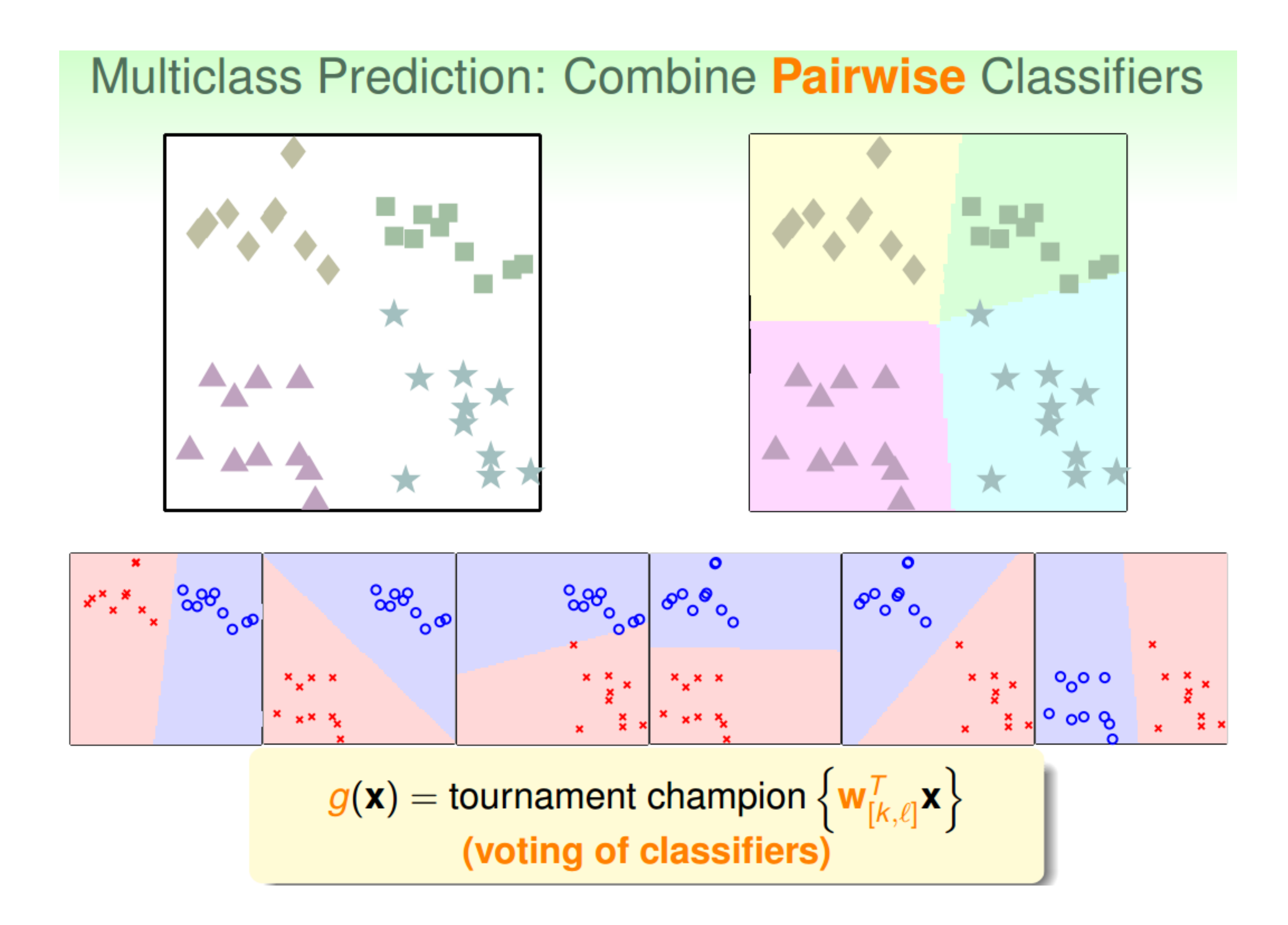
- pros: efficient,
  can be coupled with any logistic regression-like approaches
- cons: often **unbalanced** $\mathcal{D}_{[k]}$ when K large
- extension: **multinomial ('coupled') logistic regression**

OVA: a simple multiclass **meta-algorithm**
to keep in your toolbox

四、Multiclass via Binary Classification

上一节，我们介绍了多分类算法OVA，但是这种方法存在一个问题，就是当类别k很多的时候，造成正负类数据unbalanced，会影响分类效果，表现不好。现在，我们介绍另一种方法来解决当k很大时，OVA带来的问题。

这种方法呢，每次只取两类进行binary classification，取值为{-1，+1}。假如k=4，那么总共需要进行$C_4^2=6$次binary classification。那么，六次分类之后，如果平面有个点，有三个分类器判断它是正方形，一个分类器判断是菱形，另外两个判断是三角形，那么取最多的那个，即判断它属于正方形，我们的分类就完成了。这种形式就如同k个足球对进行单循环的比赛，每场比赛都有一个队赢，一个队输，赢了得1分，输了得0分。那么总共进行了C_k^2次的比赛，最终取得分最高的那个队就可以了。



这种区别于OVA的多分类方法叫做One-Versus-One(OVO)。这种方法的优点是更加高效，因为虽然需要进行的分类次数增加了，但是每次只需要进行两个类别的比较，也就是说单次分类的数量减少了。而且一般不会出现数据unbalanced的情况。缺点是需要分类的次数多，时间复杂度和空间复杂度可能都比较高。

One-versus-one (OVO) Decomposition

1. for $(k, \ell) \in \mathcal{Y} \times \mathcal{Y}$

   obtain $\mathbf{w}_{[k,\ell]}$ by running linear binary classification on

   $$\mathcal{D}_{[k,\ell]} = \{(\mathbf{x}_n, y'_n = 2 [\![y_n = k]\!] - 1) : y_n = k \text{ or } y_n = \ell\}$$

2. return $g(\mathbf{x}) = \text{tournament champion} \left\{ \mathbf{w}^T_{[k,\ell]} \mathbf{x} \right\}$

- pros: efficient ('smaller' training problems), stable,
  can be coupled with any binary classification approaches
- cons: use $O(K^2)$ $\mathbf{w}_{[k,\ell]}$
  —**more space, slower prediction, more training**

OVO: another simple multiclass
meta-algorithm to keep in your toolbox

五、总结

本节课主要介绍了分类问题的三种线性模型：linear classification、linear regression和logistic regression。首先介绍了这三种linear models都可以来做binary classification。然后介绍了比梯度下降算法更加高效的SGD算法来进行logistic regression分析。最后讲解了两种多分类方法，一种是OVA，另一种是OVO。这两种方法各有优缺点，当类别数量k不多的时候，建议选择OVA，以减少分类次数。

注明：

文章中所有的图片均来自台湾大学林轩田《机器学习基石》课程